Esame del 12 novembre 1999

Una fabbrica è composta da tre unità di produzione e diverse unità di assemblaggio. Le prime tre unità producono rispettivamente pezzi A, B e C (un solo tipo per unità), che vengono depositati in magazzino per essere poi utilizzati dalle unità di assemblaggio. Il magazzino ha una capacità limitata per ogni tipo di pezzo, rispettivamente MaxA, MaxB e MaxC, raggiunta la quale l'unità corrispondente deve aspettare che i pezzi depositati vengano utilizzati per poter continuare a produrre. Ogni unità di assemblaggio ha bisogno di un certo numero di pezzi A, B e C e la quantità dipende dal prodotto da assemblare. L'assemblaggio del prodotto può iniziare solo quando l'unità dispone di tutti i pezzi necessari.
Si descriva la sincronizzazione tra le varie unità e si implementi una soluzione usando il costrutto monitor. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

Soluzione con l'uso del costrutto Monitor:

Si suppone:
- Che i reparti producano 1 pezzo alla volta
- Che ci sia 1 reparto per ogni tipo di pezzo

La soluzione proposta cerca di utilizzare al meglio le risorse.

```
program FabbricaAssemblaggio;

type  pezzo = (A, B, C);

const MAX[pezzo] = (MaxA, MaxB, MaxC);
{ capienza massima del magazzino per ogni tipo di pezzo }

type repartoProduzione (tipo: pezzo) = process;
{ processo concorrente che produce un pezzo e lo deposita
in magazzino }
begin
      repeat
      < produci pezzo >
      magazzino.deposita(tipo);
      until false;
end;

{ processo concorrente che preleva i pezzi dal magazzino }
type unitàAssemblaggio (nA, nB, nC: integer) = process ;
begin
      repeat
      magazzino.preleva (nA, nB, nC);
      < assembla prodotto >
      until false;
end;
```

```
type GestoreMagazzino = monitor ;

var
nPezzi: array[pezzo] of integer;
{ numero di pezzi in magazzino per tipo }
codaProduzione: array [pezzo] of condition;
{ code di sospensione per i reparti produttori }
codaAssemblaggio: condition;
{ coda di sospensione per le unità di assemblaggio}
sospesi: integer;
{ numero di unità sospese su codaAssemblaggio }

procedure entry deposita (tipo: pezzo);
var i, s : integer;
begin
      { se si è raggiunta la capacità massima per questo tipo }
      while  (nPezzi[tipo] + 1) > MAX[tipo] do
            { il processo viene sospeso }
            codaProduzione[tipo].wait;

      { viene aggiunto il pezzo in magazzino }
      nPezzi[tipo] := nPezzi[tipo] + 1;
      { vengono risvegliati tutti gli assemblatori sospesi }
      { si favoriscono i processi che aspettano pochi pezzi }
      s := sospesi;
      for i := 0 to s do
            codaAssemblaggio.signal;
end;
```

```
procedure entry preleva (nA, nB, nC: integer);
begin
      { se non ci sono pezzi sufficienti }
      while  (nPezzi[A] < nA) or
                (nPezzi[B] < nB) or
                (nPezzi[C] < nC) do
            begin
                  { il processo viene sospeso }
                  { viene tenuto conto di quanti assemblatori }
                  { sono sospesi }
                  sospesi := sospesi + 1;
                  codaAssemblaggio.wait;
                  sospesi := sospesi - 1;
            end;

      { vengono prelevati i pezzi dal magazzino }
      nPezzi[A] := nPezzi[A] - nA;
      nPezzi[B] := nPezzi[B] - nB;
      nPezzi[C] := nPezzi[C] - nC;

      { vengono risvegliati i produttori sospesi }
      { poiché c'è 1 produttore per tipo, è sufficiente 1 signal }
      codaProduzione[A].signal;
      codaProduzione[B].signal;
      codaProduzione[C].signal;
end;
```

```
begin { inizializzazione del monitor }
      nPezzi[A] := 0; nPezzi[B] := 0; nPezzi[C] := 0;
      sospesi := 0;
end;

{ programma con processi concorrenti }
var  magazzino : GestoreMagazzino;
      repartoA: repartoProduzione(A);
      repartoB: repartoProduzione(B);
      repartoC: repartoProduzione(C);
      unità1 … unitàN: unitàAssemblaggio(…, …, …);

begin … end.
```

Starvation

La soluzione proposta presenta starvation nei confronti dei processi assemblatori che richiedono molti pezzi, perché vengono sorpassati da quelli che ne richiedono meno.

Soluzione 1
Si possono introdurre delle code differenziate a seconda del numero di pezzi richiesto e permettere un numero di risvegli massimo per coda prima dopo di ché si dà priorità ad altre code. Attenzione al fatto che ci sono tre tipi di pezzi.

Soluzione 2
Si può introdurre una seconda coda più prioritaria rispetto alla prima in cui vengono ri-sospesi i processi che si erano già sospesi.